The Task at Hand: Getting Airline Data In Order

Your Data

The data you'll use for this assignment are from OpenFlights.org (http://www.openflights.org).

You are provided with three data files, one for airports, one for routes, and one for airlines. The data are for up to January 2012. You'll be using this data for a couple of upcoming tasks, so be sure to keep track of them and to save your work with them.

The data in the file **airports.dat** look like this. Here are the first four (4) records in this file:

1,"Goroka","Goroka","Papua New Guinea","GKA","AYGA",-6.081689,145.391881,5282,10,"U","Pacific/Port_Moresby"
2,"Madang","Madang","Papua New Guinea","MAG","AYMD",-5.207083,145.7887,20,10,"U","Pacific/Port_Moresby"
3,"Mount Hagen","Mount Hagen","Papua New Guinea","HGU","AYMH",-5.826789,144.295861,5388,10,"U","Pacific/Port_Moresby"
4,"Nadzab","Nadzab","Papua New Guinea","LAE","AYNZ",-6.569828,146.726242,239,10,"U","Pacific/Port_Moresby"

What you have here is a character (comma in this case) separated value file.

Here are the fields in this file, according to OpenFlights.org:

- Airport ID : Unique OpenFlights identifier for this airport.
- Name : Name of airport. May or may not contain the City name.
- City : Main city served by airport. May be spelled differently from Name.
- Country : Country or territory where airport is located.
- IATA/FAA : 3-letter FAA code, for airports located in Country "United States of America". 3-letter IATA code, for all other airports. Blank if not assigned.
- ICAO : 4-letter ICAO code. Blank if not assigned.
- Latitude : Decimal degrees, usually to six significant digits. Negative is South, positive is North.
- Longitude : Decimal degrees, usually to six significant digits. Negative is West, positive is East.
- Altitude : In feet.
- Timezone : Hours offset from UTC. Fractional hours are expressed as decimals, eg. India is 5.5.
- DST : Daylight savings time. One of E (Europe), A (US/Canada), S (South America), O (Australia), Z (New Zealand), N (None) or U (Unknown).
- Tz : database time zoneTimezone in "tz" (Olson) format, eg. "America/Los_Angeles".

OpenFlights says:

"The data is ISO 8859-1 (Latin-1) encoded, with no special characters.

Note: Rules for daylight savings time change from year to year and from country to country. The current data is an approximation for 2009, built on a country level. Most airports in DST-less regions in countries that generally observe DST (eg. AL, HI in the USA, NT, QL in Australia, parts of Canada) are marked incorrectly."

The other two files, **routes.dat** and **airlines.dat**, are similar to **airlines.dat**. The fields in **airlines.dat** are:

- Airline ID : Unique OpenFlights identifier for this airline.
- Name : Name of the airline.
- Alias : Alias of the airline. For example, All Nippon Airways is commonly known as "ANA".
- IATA : 2-letter IATA code, if available.
- ICAO : 3-letter ICAO code, if available.
- Callsign : Airline callsign.
- Country : Country or territory where airline is incorporated.
- Active : "Y" if the airline is or has until recently been operational, "N" if it is defunct. This field is not reliable: in particular, major airlines that stopped flying long ago, but have not had their IATA code reassigned (eg. Ansett/AN), will incorrectly show as "Y".

Additional information about the **airlines.dat** data from OpenFlights:

The data is ISO 8859-1 (Latin-1) encoded. The special value \N is used for "NULL" to indicate that no value is available, and is understood automatically by MySQL if imported. Notes: Airlines with null codes/callsigns/countries generally represent user-added airlines. Since the data is intended primarily for current flights, defunct IATA codes are generally not included. For example, "Sabena" is not listed with a SN IATA code, since "SN" is presently used by its successor Brussels Airlines.

routes.dat has the following data fields:

- Airline : 2-letter (IATA) or 3-letter (ICAO) code of the airline.
- Airline ID : Unique OpenFlights identifier for airline (see Airline).
- Source airport : 3-letter (IATA) or 4-letter (ICAO) code of the source airport.
- Source airport ID : Unique OpenFlights identifier for source airport (see Airport)
- Destination airport : 3-letter (IATA) or 4-letter (ICAO) code of the destination airport.
- Destination airport ID : Unique OpenFlights identifier for destination airport (see Airport)
- Codeshare : "Y" if this flight is a codeshare (that is, not operated by Airline, but another carrier), empty otherwise.
- Stops : Number of stops on this flight ("0" for direct)
- Equipment : 3-letter codes for plane type(s) generally used on this flight, separated by spaces

Here's some additional information about **routes.dat**:

The data is ISO 8859-1 (Latin-1) encoded. The special value \N is used for "NULL" to indicate that no value is available, and is understood automatically by MySQL if imported.

Notes:

- Routes are directional: if an airline operates services from A to B and from B to A, both A-B and B-A are listed separately.
- Routes where one carrier operates both its own and codeshare flights are listed only once.

What You Need to Do

Here's what you need to do. You'll use Python. You can use the Enthought Canopy distribution, the Anaconda distribution, or some other version of Python.

You have three (3) data files. For each file:

1. Create labels for the fields in the files, i.e., give each field a variable name. Make your labels/names as clear and as succinct as possible. For example, you might call "Source airport ID" in **routes.dat** something like "sourceApID." Make sure that your labels are unambiguous enough that another person can understand what the variables in each file are given the above information.
2. Enter your data field labels for the file into a text file with one record, and with your labels separated by commas.
3. Read each of your field labels text files into Python.
4. Read each data file into Python as a DataFrame using the pandas package.
5. Add your variable labels to each DataFrame as the names of each DataFrame's columns.
6. Save each DataFrame by pickling it. (You'll need them later. It's good practice to check to be sure that you can successfully unpickle when you pickle.)

Your Deliverables

Provide the following in a pdf file. You can do this in few pages but try not to go over three.

1. Your variable names/labels for each data set.
2. Your Python code for:

    - Reading your variable name text files into Python
    - Reading each data file into a pandas DataFrame
    - Making the column names of your DataFrames the variable names you created and read into Python
    - Pickling each of the three data sets
    - Answering the last two questions below using the DataFrames you created

    Make sure to explain what each line of your code does.

Answer the following questions:

1. What is three letter airport code for the airport that is closest to your home?
2. How many departing routes are there from this airport?
3. How many routes are there coming into the airport with the three letter code "EGO?"

Note: You can just look up the airport to your name but the more adventurous of you might try to write code that uses the coordinates from the aiport data to compute the shortest distance to your coordinates. In fact, in the next assignment there will be an "extra credit" question where such calculations need to be made to find the longest flight routes from a particular airport. But in this assignment you can simply look up the closest airport...

Tips and Hints

Before you can use pandas you need to install it, if you haven't already done so. For example, if you using the (school) recommended Canopy IDE you would use the Package Manager to install the pandas 0.17.1 package.

You might find the documentation at (http://pandas.pydata.org (http://pandas.pydata.org)) useful in addition to what has already been made available.

Pickling is a basic Python "serializing" method. It's a way of converting Python objects in RAM to and from character streams so that they can persist. Pickle files can be text files or binary files. Here's a nice piece about pickling:

(http://python.about.com/od/pythonstandardlibrary/a/pickle_intro.htm (http://python.about.com/od/pythonstandardlibrary/a/pickle_intro.htm))

There is also some pickling actiity in the (optional) Week 2 Python Practice exercise set.

In [ ]: